REMARKS

Examiner interview summary

Attorney Katherine Wrobel participated in a telephonic Examiner interview on October 8, 2009 and proposed an amendment to the independent claims that recites the limitations of the filtering step as previously described in now cancelled dependent claims 6 and 7. Applicant respectfully thanks Examiner Kang for her time and consideration of the proposed claim amendments and for this opportunity to submit an RCE for review.

Rejections Under 35 USC 102(e)

Applicant notes that the instant office action does not restate rejections under 35 USC 102(e) that were presented in the previous office action. Applicant submitted arguments responsive to those rejections on April 7, 2008. Because the rejections under 35 USC 102(e) are not restated presently, Applicant assumes that the previously submitted amendments and remarks sufficiently overcame those anticipation rejections and that the rejections of claims 12-16, 18 and 20 under 35 USC 102(e) have been withdrawn.

Rejections Under 35 USC 103(a)

Claims 1 and 3-11 have been rejected under 35 USC 103(a) as being unpatentable over US Publication No. 2005/0114757 to Sahota et al. ("Sahota et al.") in view of US Publication No. 2003/0110253 to Anuszczyk et al. ("Anuszczyk et al"), and further in view of US Patent No. 7,107,535 to Cohen et al. ("Cohen et al."). Language on pages 3 -5 of the instant Office Action characterizes Sahota et al. as teaching every element of independent claim 1 with the following two exceptions: (1) filtering IP network event data and (2) teaching a software robot stored in the memory portion for execution by the processor portion when an end user requests playback. With regard to the filtering limitation, the Office Action provides the following language:

> Sahota does not explicitly teach that filtering the Internet Protocol network event data (page 5, 0059). However, Anuszczyk discloses such filtering was known at the time the invention was made to "reduce the amount of information transmitted (i.e. 0113)." Therefore, it would have been obvious for one having ordinary skill in the art to modify Sahota's disclosed system at the time applicant's invention was made to incorporate the teachings of Anuszczyk.
>
> Sahota further teaches automatically generating a source code from the traced and filtered Internet Protocol network event data that is executable by the processor portion- an agent spider program and generating sources from XML

> extract using XSL (i.e. page 6, 0068 and 0069). Sahota does not explicitly teach that automatically generating executable software robot that mimics the user using a web browser to access the at least one existing website. However, as described in the instant specification, the automatic code generation is performed via XSL transformation of XML (page 30). The instant specification is silent regarding to the specific implementation of such automatic code generation. Nonetheless, Sahota also discloses XSL that transforms the XML extract into other source formats would allow the XML file to be transformed in any other code (i.e. page 6, 0068 and 0069).Therefore, it would have been obvious for one having ordinary skill in the art to modify Sahota's disclosed system at the time applicant's invention was made to generate agent spider code from the XML extract by XSL for manipulation of the existing websites on behalf of users.

With regard to playback, the Office Action provides the following language on pages 4-5:

> Sahota does not explicitly teach the software robot stored in the memory portion for execution by the processor portion when an end user requests playback. However, Cohen discloses such a playback function by a web robot was known at the time the invention was made to track user activity and construct sessions of interaction with the web site (i.e. col. 4 lines 39-41). Therefore, it would have been obvious for one having ordinary skill in the art to modify Sahota's disclosed agent spider system at the time applicant's invention was made to incorporate the teachings of Cohen. The modification would be obvious because one having ordinary skill in the art would be motivated to use HTTP logs to record and track user activity and construct sessions of interaction with the web site to playback the session when a user request as taught by Cohen (col. 4 lines 39-41).

Responsive to these arguments, Applicant has amended independent claims 1 and 12 to include the limitations of dependent claims 6 and 7, which clearly define the “filtering” limitation of the present invention as comprising more than simply removing redundant information. The “filtering” process of the present invention is a packet enhancement process that comprises reorganizing, modifying, removing and adding to IP network event data so that the analyzed data takes a form that is more suited for source code generation. Applicant respectfully submits that, for the following reasons, the proposed combination of references fails to teach or suggest the limitations of independent claims 1 and 12 as currently amended.

First, Applicant respectfully disagrees with the general characterization of Sahota et al. as providing motivation for the automatic generation of source code from traced and filtered IP network event data that is executable by a processor. Sahota et al. fails to teach or suggest tracing API calls in temporal order between the system and the existing website, and tracing associated parameters and data associated with Internet Protocol (IP) network events passed to and from the API calls when a system user accesses the website. Sahota et al. thus fails to teach or suggest generating a software robot that comprises executable source code for automatically mimicking interactions between a user and a website. Applicant respectfully

disagrees with the statement that, "The instant specification is silent regarding to the specific implementation of such automatic code generation." The instant specification specifically details the tracing and filtering limitations formerly recited in dependent claims 6 and 7 and now recited in Independent claims 1 and 12. Support for these steps exists in language at paragraphs [0027], [0089] and [0092], for example. Support for the ***specific implementation*** of automatic code generation also exists at paragraphs [0094]-[0098] of the instant specification, reproduced here for convenience:

> [0094] As shown in FIG. 1, the computer software program 1 of the present invention produces source code 31 that may mimic what the user did during the tracing step 3. The source code 31 may be used on a client machine or on a server machine. Generations of source code 31 may be via an XSL transform of the XML extract data 9. Another embodiment for generating source code 31 or other useful expressions of the XML extract data 9 may be by modification of the source code 31 to produce modified source code 33. Modification may be performed in other ways including hard coding or using a computer language to parse the extract and generate some other useful product from the XML extract data 9. This optional embodiment is shown by way of a broken line in FIG. 1 between the souce code 31 and the modified source code 33.
> [0095] In an embodiment of the present invention, XSL exists for transforming the XML extract data 9 into various popular languages including JAVA, JAVASCRIPT, VISUAL BASIC, COLD FUSION, C/C++, PASCAL and many others.
> [0096] In an embodiment, parts of the automatically generated source code 31 embodying a software robot (hereafter referred to interchangeably as "source code" and "generated source code") may be integrated into a web browser as a browser plug-in 35 or extension. In an embodiment, this web-browser mode may allow interactive stepping through the generated source code 31 on a page by page basis. In the alternative, the browser mode may allow interactive stepping through the generated source code 31 on a event by event basis. In Browser mode, embodiments may allow the computer software program 1 to debug 37 the automatically generated source code 31 by displaying debug messages, HTTP header parameters, or other data in the browser plug-in 35.
> [0097] Additionally, the software robot comprising the automatically generated source code 31 may be used to perform automated page completion. In the preferred embodiment of the present invention, end-users tend to be concerned with filling in form information on multiple pages on a web site. An embodiment of the invention allows the end user to simply click on the next or similar button in the browser plug in 35 or on the actual web page, and the source code 31 sends the appropriate data over the network as if the user had actually typed in the values on the form.
> [0098] In another embodiment of the present invention, a user may click on a "run" button in the browser plug-in 35 that completes multiple forms without further user intervention.

Instead of tracing IP network event data and associated parameters passed to and from Application Programming Interfaces (APIs) making calls while a system user accesses a

website, the invention of Sahota et al. focuses on using web browsers, and in particular a web-page markup language, HTML, to generate web page layout on various platforms. For example, the abstract of Sahota et al. and paragraph [0059] describe this browser-level invention in the first sentence:

> A method and system are disclosed for acquiring and transforming existing content (e.g., Hyper Text Markup Language HTML content) for display and execution on multiple platforms and architectures….[0059] Agent spider 207 uses existing standard HTML parser engine 217 to read and transform the structure and content of any given page.

Language at page 6, paragraphs [0068] and [0069] further describes this HTML transformation by the Sahota et al. invention:

> [0068] At operation 206, the XML file or document can be transformed into a displayable format. For example, content converter 204 and content generator 203 can be used together to transform an XML file stored in XML data files 208a. In one embodiment, an XML file is transformed into an HTML web page.
> [0069] An XSL application transforms and translates XML data from one format into another. Thus, an XSL applications allows an XML file to be displayed in an HTML, PDR, Postscript or other like fomats.

Instead of tracing IP network event data and associated parameters passed to and from Application Programming Interfaces (APIs) making calls while a system user accesses a website, the invention of Sahota et al. focuses on using web browsers, and in particular a web-page markup language, HTML, to generate web page layout on various platforms. Sahota et al. teaches transforming an XML file into markup language, which is a display format language, not an executable source code. Sahota et al. specifically calls out display formats instead of programming languages. Other than a merely stating in paragraph [0059] that "modules can be easily plugged in [to an agent spider]," Sahota provides no motivation or suggestion as to how to acquire the data within any given module and, particularly within the context of display format transformation, Sahota et al. makes no suggestion either implicitly or explicitly for generating source code. The invention of Sahota et al. thus focuses on web page layout and markup language detailing that layout, instead of tracing IP data packets to and from API calls on the Internet Protocol network level of system architecture (for example, the TCP/IP and OSI models) to produce a software robot comprising executable source code.

Unlike the present invention, Sahota et al. fails to teach or suggest generation of a software robot that comprises executable source code that mimics user interactions with an existing website. Despite this fundamental different between the invention of Sahota and the

present invention, applicant has modified independent claims 1 and 12 in an effort to advance prosecution and further highlight these fundamental distinctions over Sahota et al. Applicant agrees with the Examiner's statement with regard to Independent claim 1 that "Sahota does not explicitly teach that filtering the Internet Protocol network event data (page 5, 0059)." With regard to claims 6 and 7, however, the Examiner recites portions of Sahota et al. as teaching these filtering steps. Applicant has moved the content of dependent claims 6 and 7, which detail the packet enhancement type of filtering of the present invention, into the Independent claims. Because Sahota et al. fails to teach or suggest tracing IP network event data and automatically generating a software robot as taught by the present invention, Sahota et al. also fails to teach filtering as that term is defined in the present application. With regard to dependent claim 6 for example, the instant office action provides the following language:

> Per claim 6:
> Sahota further discloses:
> - filtering is adapted for removal of redundant and useless IP network event data passed to and from the API calls (i.e. page 7, 0076):
> - removing network management packets that are acknowledgements and retries (i.e. page 9, 0103)
> - collating IP packets into single HTTP based messages; and collating HTTP based messages into single records of content objects, wherein the content objects comprise HTML, images, audio, and other HTTP content (i.e. page 3, 0039; page 9, 0099,0103, 0104, 0108; page 10, 0115).

With regard to "filtering" Sahota et al. is concerned with filtering content instead of filtering low level network protocols. For example, language at paragraph [0076] of Sahota et al. highlights this distinction:

> [0076] Syndication server 110 then transforms, e.g., the HTML web page, from web server 112 into syndicated content for the set-top browser 108 running on TV 104. In one embodiment, content harvest and conversion platform 130 performs the transformation process using software modules, which may be executed within the harvest and conversion platform 130 or within syndication server 110. Syndication server 110 can also perform such functions, which include caching web pages, storing web pages in a data base, consolidating diverse source feeds into an XML file or an HTML web page that is sent to set-top box 106, filtering information, or tracking usage.

Anuszczyk et al. fails to cure the deficiencies of Sahota et al. with regard to teaching or suggesting filtering in the process of automatically generating source code as claimed in Independent claims 1 and 12 as currently amended. Anuszczyk et al. references filtering within the context of monitoring a network to identify elements therein, such as hardware and programs stored on servers.:

> [0067] The system and method described herein can discover components in the IT system, determine dependencies between the components, generate a visual map of the components in the IT system, and then track changes to the discovered components and the dependencies between the components. The first step in one embodiment, therefore, is the discovery of components in the IT system, which determines what is installed and where it is installed.

In order to create a visual map of an IT system, the invention of Anuszczyk et al. monitors "events," defined at paragraph [0030] as, "…occurrences in the IT system, such as file or registry creations, modifications, or deletions, or the discovery of such components in the IT system." The "filtering" of Anuszczyk et al. concerns sorting these events to reduce the amount of information transmitted. Only events related to a particular component are transmitted:

> Generally, each filter used in the IT system should let through the event information that matters for discovery and tracking, and filter out the remaining event information. The event information that matters, generally, is event information that corresponds to elements of a fingerprint or that corresponds to a detected or discovered component.

Again here, no tracing and filtering of IP network layer data packets occurs. By comparison, the present invention is not just discarding data to reduce data, but is reorganizing, modifying and adding temporal order to the IP network level data stream. Therefore, "filtering" of the present invention is effectively packet enhancement and preparation for generation of source code. The present invention traces a temporal sequence of IP network layer data packets transferred to and from API calls while a user is accessing an existing website from a computer processing executable steps. This IP network layer tracing and filtering steps are a critical components with which the present invention generates source code. Anuszczyk et al. provides no motivation or suggestion for creating these steps as claimed in independent claims 1 and 12 as currently amended.

Cohen et al. also fails to cure the deficiencies of Sahota et al. taken alone and in combination with Anuszczyk et al. Like Sahota et al. and Anuszczyk et al., Cohen provides no teaching or suggestion of the automatic generation of source code. Cohen et al. monitors and collects usage data related to a website by tracking user clicks on certain parent and child pages during a user session.

> Using pre-programmed basic comparison rules and computer based mathematical models, matrices are used to represent statistical information about the visitor's sessions on the web site. The statistical information is used to extract visitor behavior which was unexpected (anomalies). Anomalies are grouped into recommendations. These recommendations are used to automatically customize the web site. In the alternative, information is provided to the web site

> administrator to customize the web site to be more efficient and visitor friendly, maximizing the operation of the Web site and promoting more frequent visits.

Cohen et al. is concerned with producing a log of these user selections during a session so that a web content creator may modify layout and improve efficiency and display appeal of the website based on statistical data about user interaction with a website. For example, text at column 4, lines 39-50 and column 5, lines 34-49, reproduced here in pertinent part for convenience, describe this log creation:

> One embodiment of the invention uses HTTP logs to record and track user activity and construct sessions of interaction with the web site… In another example, suppose the actual user session was object A 302→object B 304→ object A 302→ object C 306. In this exemplary embodiment in which HTTP logs are relied on to reconstruct the sequence of accesses for each user, this user session may be recorded by the HTTP logs as a session: object A 302 → object B 304 →object C 306.

Cohen et al. provides no motivation or suggestion for automatically generating source code based on tracing and filtering IP network level data. Because Cohen et al. provides no motivation or suggestion for generating an executable software robot that mimics the user using a web browser to access the at least one existing website, Cohen et al. therefore also fails to teach or suggest any playback feature. Instead, Cohen et al. teaches the creation of matrices that enable statistical anlysis of website usage. An analyst may study this data and elect to manual produce code that repurposes website content inaccoradnce with data analysis, but Cohen et al. fails to suggest any IP network level data analysis at all, much less IP network level data analysis that enables the automatic generation of source code.

Sahota et al. teaches transforming one markup language, which is a display format language distinct from an executable source code, to another markup language (i.e., transforming Extensible Markup Language (XML) into Hypertext Markup Language (HTML)). The invention of Sahota et al. appertains to web designers, people of ordinary skill in the art of content transformation. In contrast, the present invention appertains to programmers capable of essentially reverse engineering a website through analysis and interaction with the protocol layer to produce a sophisticated system that interacts with a website. Sahota et al. does not motivate or suggest the transformation of a markup language to executable source code. One skilled in the art of software robots comprising automatically generated executable source code would not look to the web browser invention of Sahota et al. either alone or incombination with the network protocol tracing invention of Anuszczyk et al. and the network usage data collection invention of Cohen et al. to teach, motivate or suggest tracing and IP data packets on the

network level and selectively removing, collating and analyzing (i.e. filtering) those IP data packets to generate an extract file and automatically generate a software robot that mimics user interactions with a website.

Applicant respectfully submits that because the proposed combination of references fails to teach or suggest the limitations of Applicant's independent claims 1 and as currently amended, independent claims 1 and 12 are in condition for allowance. Because claims 3 through 11 depend from independent claim 1 and include all limitations thereof and because claims 13 and 15 through 20 depend from independent claim 12 and include all limitations thereof, applicant respectfully submits that those dependent claims are also in condition for allowance. Applicant respectfully requests reconsideration and withdrawal of the present rejection.

Claims 12, 13, 15, 16, 18, and 20 have been rejected under 35 USC 103(a) as being unpatentable over Sahota et al., in view of Cohen et al. and further in view of Anuszczyk et al. As described above in detail, Applicant respectfully submits that the proposed combination of references fails to teach or suggest the present invention as claimed in independent claims 1 and 12 as currently amended. Both independent claims now recite the details of now cancelled dependent claims 6 and 7 which particularly describe the "filtering" of IP network event data. As described above, the proposed combination of references fails to teach or suggest the automatic generation of source code and, in particular, fails to teach or suggest the tracing and filtering elements of the present invention.

Applicant respectfully submits that because the proposed combination of references fails to teach or suggest the limitations of Applicant's independent claims 1 and as currently amended, independent claims 1 and 12 are in condition for allowance. Because claims 3 through 11 depend from independent claim 1 and include all limitations thereof and because claims 13 and 15 through 20 depend from independent claim 12 and include all limitations thereof, applicant respectfully submits that those dependent claims are also in condition for allowance. Applicant respectfully requests reconsideration and withdrawal of the present rejection.

Claims 17 and 19 have been rejected under 35 USC 103(a) as being unpatentable over Sahota et al. in view Cohen et al, further in view of Anuszczyk et al and still further in view of US

Patent No. 7,231,606 to Miller et al. ("Miller et al."). Because these claims depend from independent claim 12, this response will address the present rejection on the independent claim level. As described above the proposed combination of references fails to teach and suggest independent claim 12 as currently amended. Namely, the proposed combination of references fails to provide any motivation for teaching the automatic generation of source code based on tracing, filtering and analyzing IP network event data to produce a software robot that mimics user interactions with a website.

Miller et al. fails to cure the deficiencies of Sahota et al. in view of Cohen et al and further in view of Anuszczyk et al. The newly proposed combination of references fails to teach or suggest independent claim 12 as currently amended. Specifically, independent claim 12 teaches the automatic generation of a software robot comprising source code that mimics interactions between the system user and the existing website wherein the software robot is adapted for playback on the IP network level at the request of an end user. In contrast, Miller et al. teaches creating a test script for testing a website via a test-enabled Internet browser for ensuring repeatable results of the website under test. Column 2, lines 59-57, reproduced here in pertinent part, describe the invention of Miller et al.:

> According to a general aspect of the invention website testing is performed in a browser environment….The invention thus can be implemented in a test-enabled Internet browser.

Miller et al. teaches a user-directed recordation of webpage results to create a user-defined script for testing a webpage at the test-enabled web-browser level. For example, column 3, lines 21-25 and lines 56-59 describe this high level of user involvement in the recordation of the test script:

> Here, a test-enabled web browser is used in the normal way to interact with the website under test. During this interaction, the user can command certain kinds of data to be extracted into a script file as a part of the recording process…During recording a user can select and specify certain information that is extracted immediately from the current rendition of a web page and which is deposited in special formats within the recorded script.

Not only does Miller et al. focus solely on the web browser level, but the reference also fails to teach the steps of tracing, filtering and analyzing IP network data and parameters passed to and from API calls to automatically generate source code comprising a software robot. Moreover, like Sahota et al., Miller et al. fails to teach automatically generating any software robot. Miller et al. teaches away from automatically generating a source code by tracing, filtering and

analyzing IP network event data and instead teaches creation of a script recorded through user commands during interaction with a test-enabled web browser. Those user commands are vital to practicing the invention of Miller et al. and providing a means for validating the website under test.

Further in support of this distinction is that the invention of Miller et al. is limited in scope to recording user-selected events in the Internet Explorer test browser, or to recording windows operating system function key presses on a keyboard and mouse clicks. Miller et al. fails to teach, suggest or provide any motivation for creating executable steps enabling network tracing for automatically generating a software robot. For example, language at column 9, lines 7 through 23 and lines describe the focus of Millers et al. as centering entirely on a test-enabled web browser:

> 13. Record/Play--Application Mode
> During the recording process the user can signal a change in internal recording state using the control GUI from the normal recording mode to Application Mode recording. This mode has the advantage that it can apply not only to activity within the browser window but also to any other application launched by the test enabled web browser that exists on the Windows desk-top. In application mode recording only keyboard activity, mouse click, and mouse drag activity is recorded. In application mode the only validation modes are a partial image synchronization and validation of text that is selected by the user and put into the clipboard, a logical area that is maintained by the underlying Windows environment, that contains the results of a COPY instruction.

Here, Miller et al. describes a “normal mode” which performs browser based recording, and “application mode” in which the invention records only keystrokes and mouse events. No suggestion is made of recording network level tracing or how to proceed with such IP level network tracing.

Language at column 9, lines 40 through 67, reproduced here in pertinent part, further describe the teaching of Miller et al. as focusing solely on WINDOWS screen events:

> 15. Record/Play--Multiple-Window (Sub-Browser) Operation
> In accordance with the representative environment the special web browser must test WebSites that evoke multiple windows reliably… In the representative environment this is accomplished by automatically opening a second instance of the testing browser, one already designated to be in record mode, whenever the parent page requests a new window. In the representative environment resulting sub-script is recorded in an script file named automatically and systematically by eValid according to the sequence number of the sub-browser that was launched and the depth of nesting of the recording sub-browser… During playback the representative environment automatically launches a copy of the special browser in response to the LaunchSubBrowser command and instructs the launched sub-

> browser to play back the indicated sub-script. This accomplishes the effect of simultaneous parent and child playbacks exactly as was recorded initially.

Here, Miller et al. clearly fails to teach network based tracing/recording with regard to the recording of interactions of "sub browsers". Miller et al. teaches away from network based tracing by explicitly launching sub browsers during playback of the script rather than by directly generating IP network calls to perform the playback.

Miller et al. clearly teaches away from tracing IP network events and automatically generating a source code adapted for playback at the network level upon request by an end user. In contrast, the present invention traces events directly at the network layer and therefore enables playback in the absence of a browser or other application to simulate user interactions with a website. Running in a web browser as a plug in is only one option for playback of the automatically generated source code of the present invention and is not the only option. For example, at least FIG. 1 and Specification paragraph [0099], reproduced here in pertinent part, describe the ability of the software robot of the present invention to run either as a web browser plug in OR to run as an independent source code operating independent of a web browser:

> [0099] FIG. 3 illustrates the relationship of various software modules of the computer software program 1 in an embodiment. The software modules may be divided into those that run during the analysis data capture phase, and software that may run either on a client and/or on a server machine during playback of generated source code 31. Client playback may be either via a dedicated win32 application 34, or a web browser plug-in 35 and/or browser helper object. [Emphasis added]

Because Miller et al. fails to cure the deficiencies of the earlier proposed combination of references, the newly proposed combination of references also fails to teach or suggest every element of independent claim 12 as currently amended. Applicant respectfully submits that independent claim 12 as currently amended is in condition for allowance and that dependent claims 17 and 19, which depend from claim 12 and include all limitations of that independent claim, are also in condition for allowance. Applicant respectfully requests reconsideration and withdrawal of the present rejection.

Summary

In light of the above, Applicant respectfully requests consideration of the subject patent application. Any deficiency or overpayment should be charged or credited to Deposit Account No. 50-4514.

Respectfully submitted,

/Kevin M. Farrell/
Kevin M. Farrell
Attorney for Applicants
Registration No. 35,505

October 13, 2009
Pierce Atwood, LLP
One New Hampshire Ave., Suite 350
Portsmouth, NH 03801
603-433-6300